How to Ingest Data into Google BigQuery using Talend for Big Data

A Technical Solution Paper from Saama Technologies, Inc.
July 30, 2013

How to Ingest Data into Google BigQuery using Talend for Big Data

In this post, we will examine how the Talend Big Data Integration tools can be used effectively to ingest large amounts of data into Google BigQuery using Talend for Big Data and the Google Cloud Platform. We also propose deployment architecture for Talend for Big Data on Google Compute Engine, which can scale with the data volumes and SLA requirements for data load windows.

Intended Audience

This post is fairly technical and is intended for architects, and tech-savvy analysts who are exploring ways to use Google BigQuery for their Big Data analytical needs, and want to automate, scalable data ingestion using standard ETL/Data Integration tools.

What you will Learn

This is a real experience implementing a scalable solution for a client of Google. After going through this post, you should able to

1. Develop fair understanding about how Talend for Big Data can be used for data ingestion into Google BigQuery
2. Understand how to scale Talend for Big Data deployment to manage large amounts of data processing and ingestion into Google BigQuery
3. Have a conceptual understanding of the challenges and limitations of using Talend ETL to ingest data into Google BigQuery.

Background

Saama Technologies, Inc. is a pure-play data science solutions and services company, focused on solving the data management and advanced analytics challenges of the world's leading brands. Saama is a strategic implementation partner for the Google Cloud Platform division,

helping Google customers with their Big Data implementation, strategy, architecture development, as well as ongoing operations of implementations. This post is based on an implementation experience where we were tasked with developing an end-to-end data ingestion pipeline and incremental data loads with specific requirements.

Solution Scenario

Big Challenge: Retailer Needs Fast Access to Big Data and Shorter Cycle Time to Information

A large U.S. retail Google Cloud Platform customer wanted to load several terabytes worth of data into BigQuery for performing Point of Sale (POS) and Market Basket analysis. Specifically, they needed to quickly analyze the number of household trips for profitability during the busiest and most profitable hours within a given store for any selected product or brand. As a part of this project, five objectives were established, mainly associated with data loading, processing, and querying.

1. Load approximately 6 TB of data representing about 36 months worth of data (close to 2 Billion transactions) under 10 hours (on time)
2. Incremental data loads (less than 200GB per batch) will happen through data being uploaded directly to Google Cloud Storage
3. Data pre-processing and transformations to be performed before/during data load.
4. Use of cloud infrastructure for most of the tasks involving data processing and loading.
5. Complete lights out operations.
6. Solution should scale as data volumes grow (tens to hundreds of terabytes in future).

We designed and built a solution using Google Cloud Platform and Talend for Big Data. It has been designed to scale at much larger volumes and lower load windows.

This data was used downstream by business analysts and data scientists for meaningful and valuable analytics. For this paper, the type of analysis and visualization that was delivered is not a focus.

Our Approach to Scalable Data Ingestion Architecture

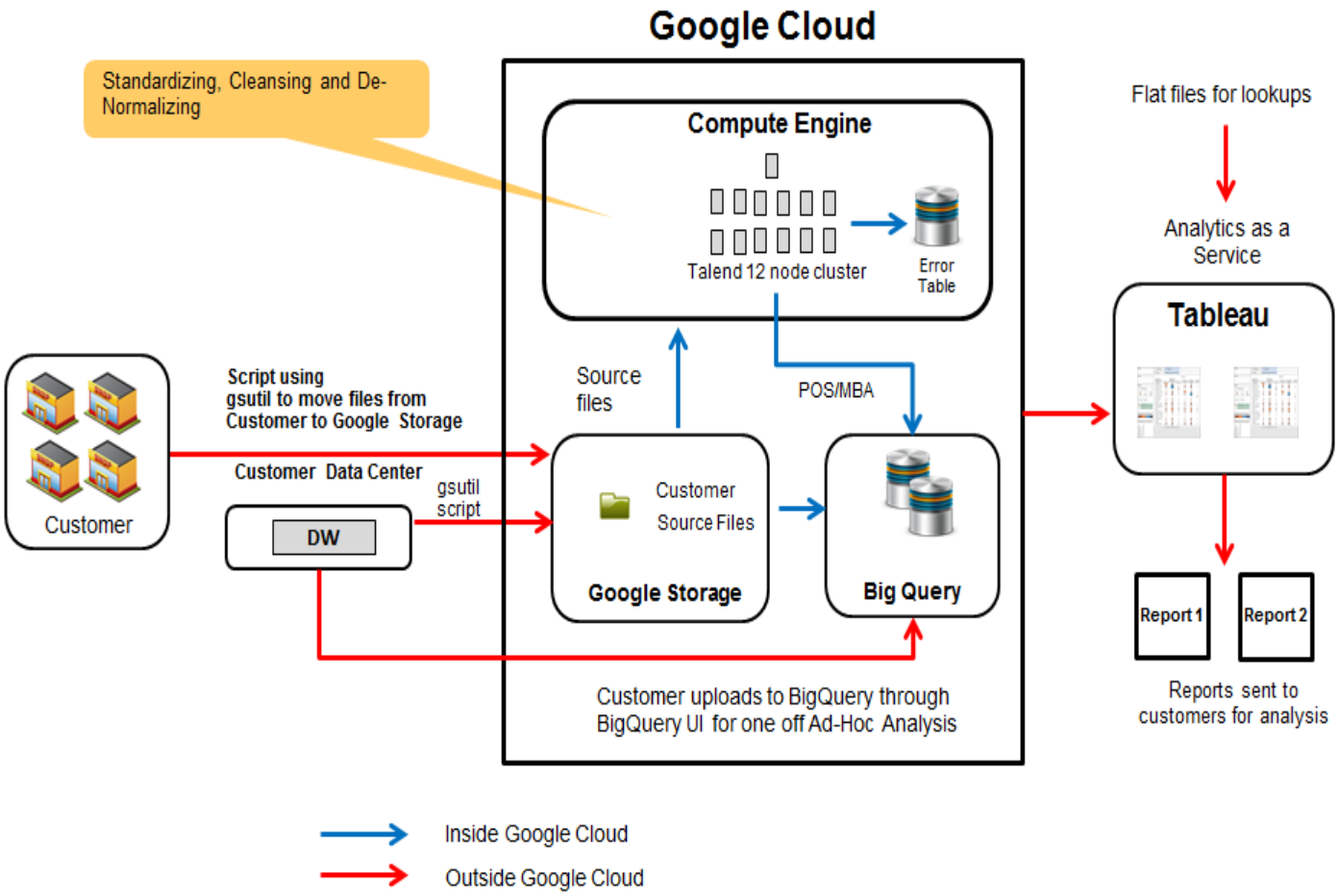
Our Big Data architects worked closely with Google Cloud Platform experts, Talend experts, and referred to best practices to provide high impact business insights, for a state-of-the-art scalable big data infrastructure, which can process massive amounts of data in a timely fashion.

We chose Talend for their data integration capabilities which included native connectivity to Google Cloud Storage and Google BigQuery. While Talend was not immediately available on Google Compute Engine, we were able to work with Google Compute Engine and highly responsive team at Talend to make some tweaks and get Talend multi-node deployment to work on the Google Compute Engine.

We made extensive use of the utilities provided by Google to manage the data transfer into the Google Cloud Storage. These python based utilities are versatile and can be easily scripted in with proper error handling.

The result was an architecture that could scale on demand, process the data sets with ease, and can drive lights out operation because of the inbuilt error handling.

Data Ingestion Architecture



(Diagram 1.1)

Below are the details of the components used in the data ingestion architecture.

Google Cloud Storage

Google Cloud Storage buckets were used to store incoming raw data, as well as storing data which was processed for ingestion into Google BigQuery. Below are the details around the incoming raw data:

1. Monthly data files (approximately 100-200 Million records) for transactions for the duration of 36 months.

2. 5 Master data files. These files were static and changed only once every few months.
3. Approximately 6TB worth of data.

Google BigQuery

We used Google BigQuery (release 1.3) for hosting processed data and providing analytical services.

Google Compute Engine

Google Compute Engine enables the running of large-scale computing workloads on Linux virtual machines hosted on Google's infrastructure. We used the Google Compute Engine for running an ETL workload with Talend for Big Data. For our particular case, we used 12 nodes. These nodes can be added on demand based on the workload. The following are the details of the master and slave nodes:

Node Type	Quantity	OS	# of Cores	Memory	Local Disk Storage
Master Node	1	CentOS 64 BIT	8	64GB	2 TB
Slave Node	11	CentOS 64 BIT	8	64GB	210 TB

Google Python Application gsutil

Google provides a couple of Python applications which are essentially command line interfaces to API. These applications enable command line access to Google Cloud Storage (gsutil) as well as to Google BigQuery (bq). We mainly used gsutil which is a Python application that lets you access Google Cloud Storage from the command line. You can use gsutil to do a wide range of bucket and object management tasks, such as object copy, creation, deletion, etc. We used gsutil for transferring raw data into Google Cloud Storage buckets. Data transfer tasks were scripted using Python scripts for uploading data.

Tableau Data Visualization - Desktop

We chose Tableau 8.0 for data visualization. Data visualization and analysis is not part of the scope of this paper.

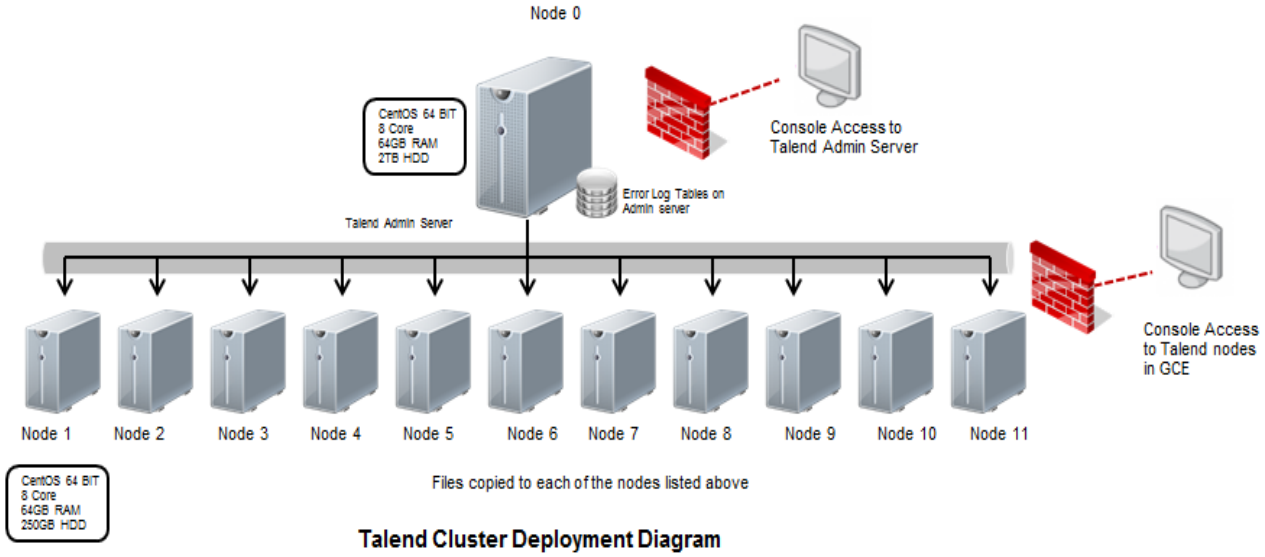
Talend for Big Data

Talend Platform for Big Data (v5.2.2 part of the Talend for Big Data product family) was utilized for this project, for which the following components had to be installed:

- Master Node:
    - Talend Administration Center - Admin Console to administer the enterprise cluster
    - Talend Studio - Used to create the Source-Target mappings
    - Job Server - Execution Server
- Each Slave Node:
    - Job Server - Execution Server

Talend for Big Data Deployment Architecture

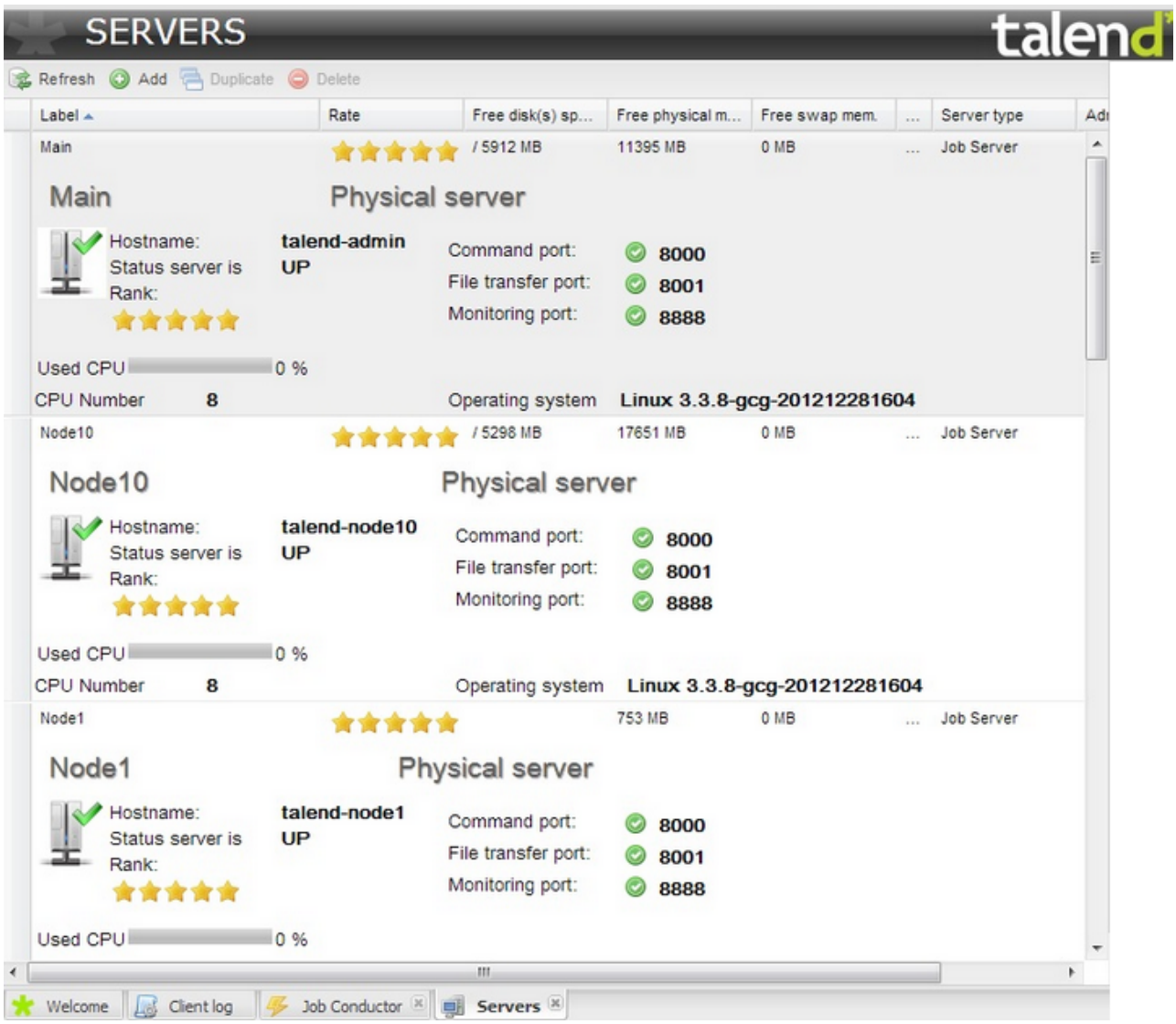
Talend for Big Data has been designed to work in large-scale data processing workloads through massive parallelization. It is very easy to use the central administration capability which enables fast deployment and monitoring of nodes, and massive parallelization. Talend for Big Data also provides a connector to Google BigQuery making it easy to ingest processed data through multiple nodes into Google BigQuery.



Talend Cluster Deployment Diagram

In order to perform parallel processing efficiently, we made copies of reference data on each node, and copied uniquely shared transaction data on each node. Master mappings representing necessary data transformation were distributed from the master node onto each of the slave nodes. This enabled us to cut down processing time on all the data by 300% to under 8 hours.

We created a preconfigured image of a Talend slave instance. This was used to create slave nodes during the deployment. In order of installation, the administration console is installed first followed by installation of each of the individual nodes. While installing the Talend nodes, Talend requires the user to enter details of the admin interface. The nodes can be administered from the administration console as depicted below:



Processing and Loading Data into BigQuery using Talend for Big Data

In this particular example where we designed scalable BigQuery data ingestion process using Talend for Big Data, we had following data to process:

- Master Data, such as Customer/Prospects (about 30 Million records), Product/SKU information (about 2 Million records) and Location and Org data (about 2000 records).

- Transaction data, such as header, details and payment information (**~9 Billion records, eventually de-normalized to fewer records**)

Based on specific query patterns, as well as requirements around how data will be accessed by end users, we made a couple of decisions about how we wanted to organize data, and we decided to:

- De-normalize data and store it as one single data set, hence we required that we joined transaction data with master data and resolved all master data references before loading data into BigQuery
- Sharded transaction data by year (based on data query patterns, manageability and cost considerations, etc.). We ultimately created 3 denormalized tables, one representing each year to which the data belonged.
- As a part of the data processing step, anytime we could not resolve the master data reference to a transaction, we captured those transactions in an error table for further processing.

Implementing Transformation Logic using Talend

For achieving necessary scalability, we had envisioned deployment of ETL and the transformation/processing jobs to be done across several nodes (master/slave configuration). This meant that during transformation development, we assumed that access to necessary data (files) be available from each node with basic configuration.

Step 1: Creating Source and Target Connections

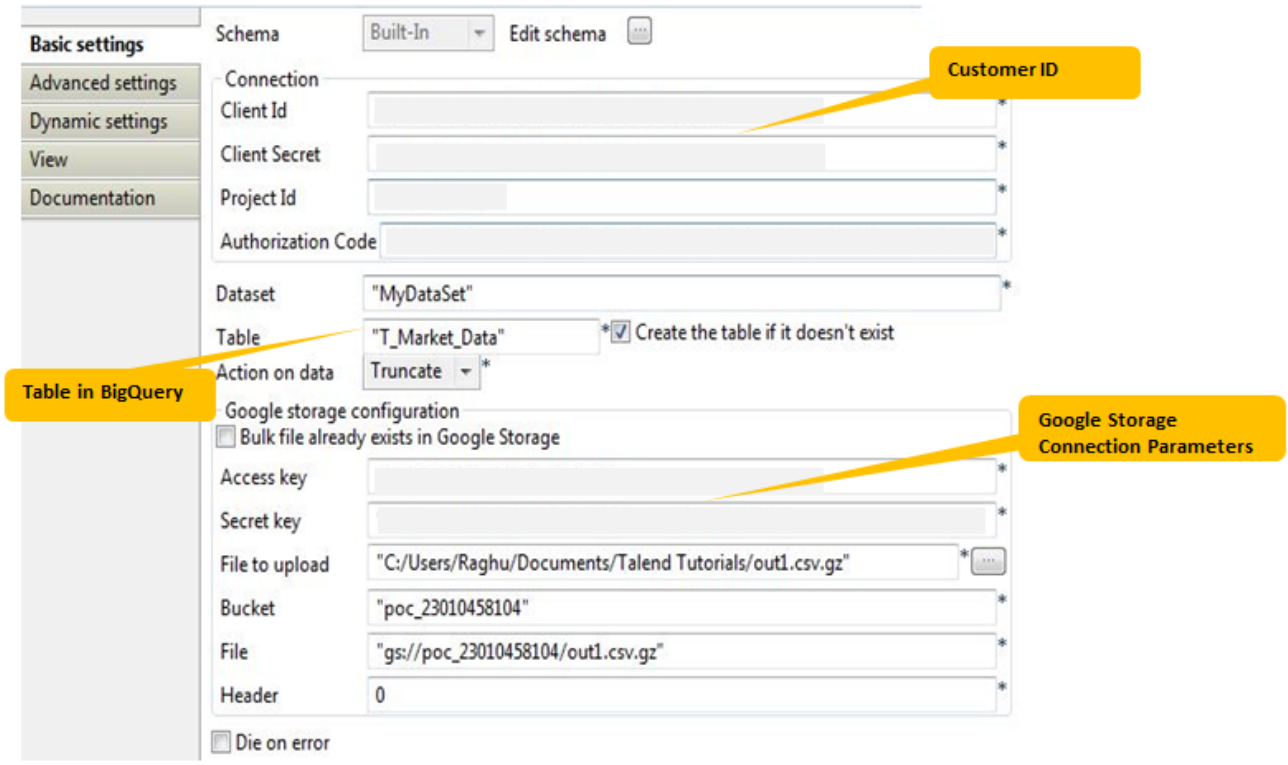
Source:

Using a Talend flat file connector type, we created connections to the 7 source files (refer file types in Transformation Logic section) representing master and transactional data. We split the gigantic file into multiple files for distribution across nodes, since we are in the process of

defining metadata for transformation, we don't need to identify each individual file we will be loading here.

Target:

Using the Talend *tBigQuery* component, create a target connection to BigQuery. This step just creates a connection to your BigQuery project, and does not specifically create the target table yet.



Step 2: Creating Source & Target Definitions

Source Definition:

This step lets you define or modify the metadata of the source file. Within this step you can modify the data types for each of the columns of the source files

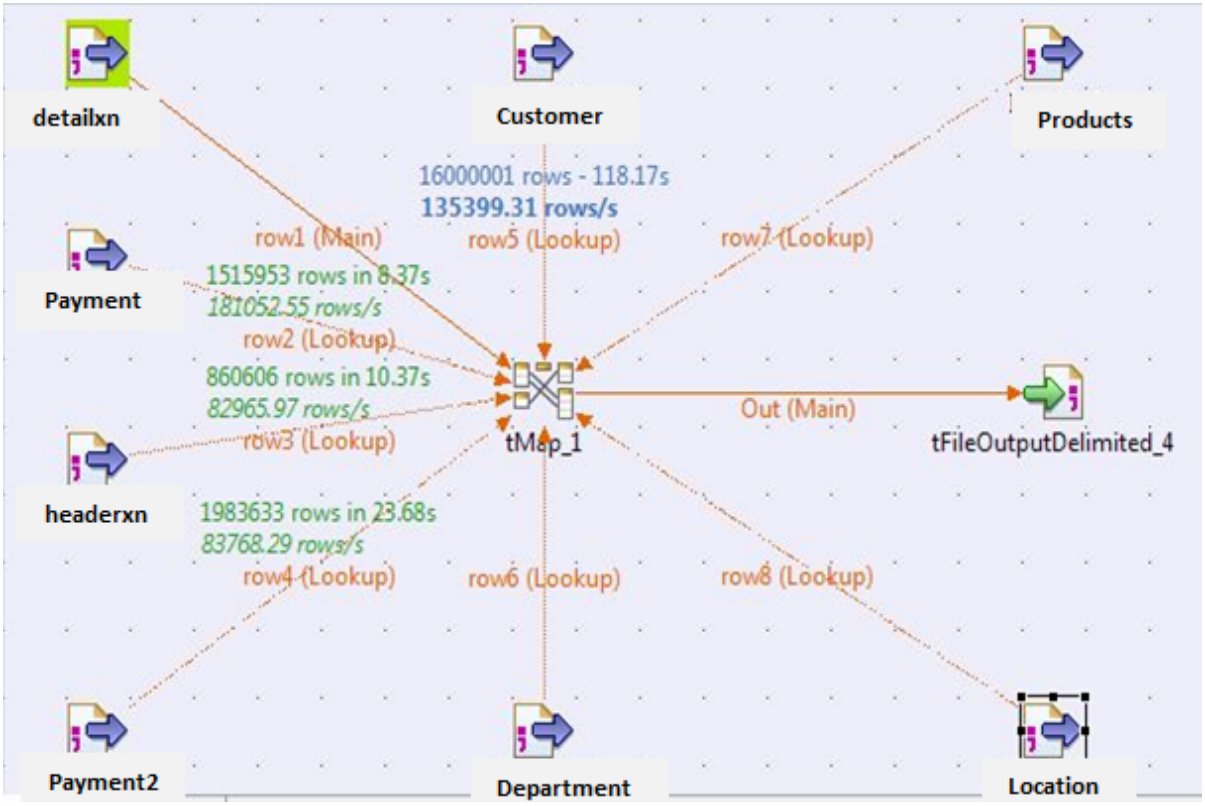
Target Definition:

This step lets you define the target table structure in BigQuery, that lets you define data types and size of each of the columns. This way any corrections or modifications to the data types can be done to meet BigQuery data types compatibility.

Step 3: Create Transformation Mapping

Talend provides you with an easy to use graphical user interface, where you are allowed to select your source and target tables and create a mapping. A mapping is nothing but a set of rules to transform data from source to target. Talend has a map component that lets you create a mapping.

In our use case above, we had 7 source files that were merged into one target table in BigQuery.



The transformation logic in our case was straightforward - we just needed to comply with certain data types that BigQuery supports. As an example, BigQuery does not support LONG datatype, so we had to transform a LONG to the FLOAT datatype. In addition we had to perform some String to Timestamp conversions (to transform incoming strings representing dates into Timestamp data types within BigQuery), to ensure time series analysis was easily possible in BigQuery.

Running Talend ETL Jobs to load data in BigQuery

For performing initial data load we had to make sure that data was ready and accessible to all the nodes for Talend server. Below are the detailed steps we had to take as part of initial data load:

Step 1: Source Files Readiness

To speed up the loading process, the transactions file which contained three years of transactions was split into smaller files that consisted of a single day's transaction. This allowed us to process the loading in parallel for each day from across all available nodes. Python scripts were used to split the files on the master node. The split resulted in 365 sales transaction headers, sales transactions, and payment files.

Step 2: File Distribution across Nodes

We used 11 slave nodes of Talend configuration for parallel processing and the set of 365*3 transaction files were evenly distributed across the 11 nodes. The master data files were distributed ‘as is’ across each of the nodes. Python scripts were used to distribute the files across the nodes

Step 3: Executing the Mapping

Talend saves each of these mappings as .JAR file within its workspace. If a user wants to run the same mapping from multiple nodes over a set of split files, the user needs to deploy mapping .JAR files across the different nodes through the Talend administration console. Each of the nodes now has a local copy of the mappings file and will refer to a specific location on the node to get parameterized source files.

From within the Talend Administration console, an admin can now begin running the mapping on a single node or simultaneously on multiple nodes, and can monitor the progress of each of

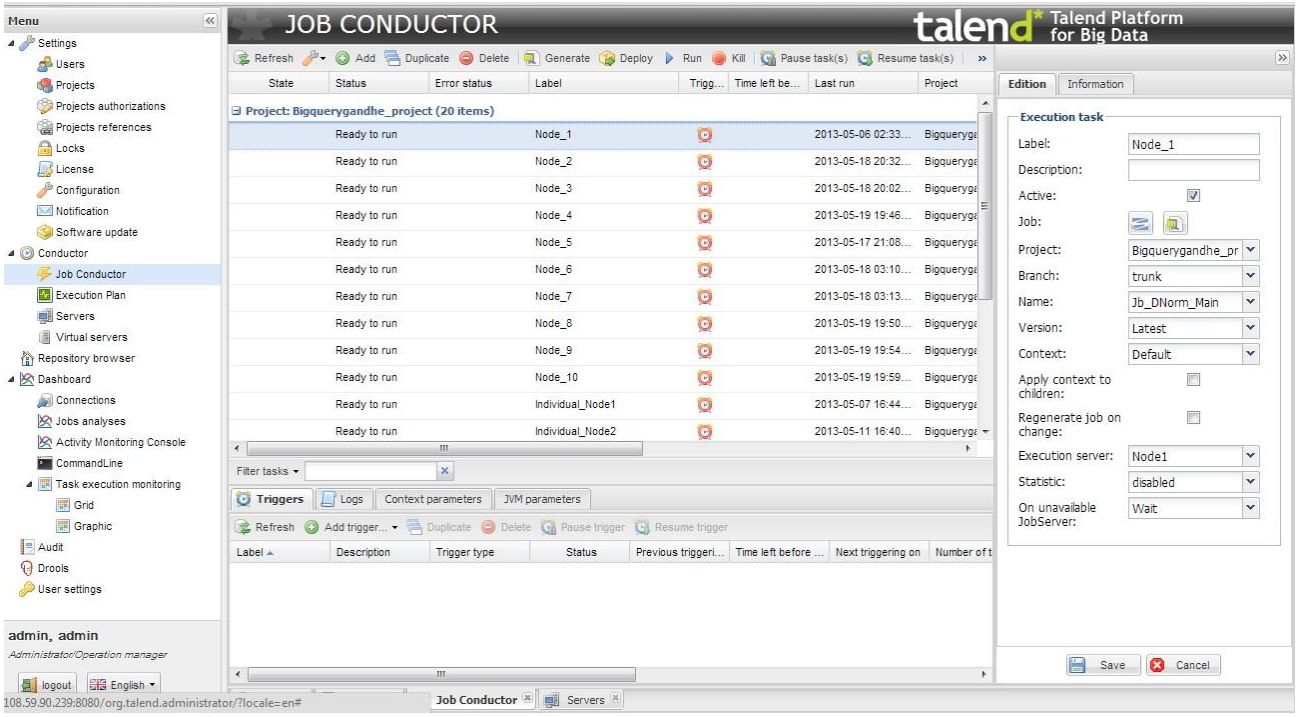
the jobs in a graphical user interface. Our use case required us to run these jobs over three year datasets, one year at a time over eleven Talend nodes.

On our first run, a single node took us four hours for an individual month of sales transactions. On further optimizing the lookup logic (loading the dimensions into memory), the overall processing time for a mapping to get completed for a single month's worth of data had been reduced to two hours. After our initial validation, we deployed the mapping over all 11 nodes and used an entire year of data (~1.6 Billion transactions) split into multiple files on each node. The Overall data load process for an entire year of data took merely 2.25 hours. This was a significant achievement as we were in a position to process three years of data in about seven hours meeting the customer SLA.

The entire process was scheduled and automated to an extent where an admin just needs to trigger the jobs in one place and monitor the progress.

Monitoring the Jobs within Talend Master Node

Since we used a Master/Slave configuration of Talend to parallelize the loading process, it was easier to monitor the jobs on slave nodes from the master node console. The following is a sample screenshot which shows how this monitoring can be done. The real advantage of this configuration is that the administrators need to login to only one screen for monitoring all the nodes.



Data Validation

Once the job for a year is completed, we created a standard set of data validation queries to validate the data pre-transformation with the data post-transformation. Any exception identified was addressed on a case-by-case basis. We also looked at all transactions which were captured in the reject table for further actions.

Outcome and Lessons

There were many lessons we learned about each and every technology component used in this architecture. Below is a brief summary of what we learned from our experience. If you would like any further information or have any questions or need to discuss any of the configurations in detail, please do not hesitate to **reach me at bigdata_questions@saama.com**

Customer Problem Solved

Our Saama team was able to demonstrate the power of the Google Cloud Platform, transforming terabytes of fast moving data into insights *within seconds*. The Google Cloud, Talend, and Saama Technologies solution gave the retailers' data scientists an option to ask on-demand business questions, and scientists were productive in days as opposed to weeks or months, as the wait time for information was reduced to seconds.

What We Learned

Google Cloud Platform is highly scalable and very easy to use. We were able to stand-up an environment which was required to process and ingest large amounts of data in mere minutes, and were able to build a highly scalable solution which can be used for any data volumes. Google BigQuery is really a revolutionary concept when it comes to analyzing large amounts of data. Data ingestion within BigQuery using Talend was a breeze.

Talend for Big Data worked very well. Our Architects who have had experience dealing with multiple ETL tools in the past were able to easily architect Talend for a multi-node parallel data ingest. Master-Slave Orchestration across all 12 nodes worked flawlessly. We were able to get Talend to work on the Google Compute Engine, thanks to fantastic support from the Talend support teams.

Challenges

As with any project or initiative, there are challenges one faces. While developing this scalable pipeline for data ingestion, we also had some issues. We were able to overcome most of these issues but we thought we would document those here for the benefit of readers.

Parallel Load Execution

When establishing a connection with BigQuery for data inserts, keep in mind that currently BigQuery limits such concurrent connections (for data inserts only) to 20. This poses a limit on the number of parallel jobs/sessions you can run through Talend at any given time.

The way to optimize available capacity to accommodate this limit is to monitor the number of jobs running, and have a new job start immediately when there is an open slot for a job (within the 20 limit).

Configuration of Google Utilities

Installation of Google utilities is very straightforward. We had to install these utilities at multiple locations, including the client locations for seamless data transfer to Cloud Storage. The firewall/proxy configurations at the client site presented a challenge for us. To solve the proxy issue, we had to define the proxy configuration in the .boto file for gsutil. We still had another issue with the firewall blocking some SSL certificates so we had to get help from experts. Again, some of the firewall blocking issues will be different for different sites and you may not encounter these challenges.

Miscellaneous

Many of the miscellaneous issues were addressed upfront as we went through validation of data quality and standard checks. For example, when ingesting date type fields, BigQuery expects data to be in a very specific format - if you don't comply with it, records get rejected. Also, BigQuery expects that records in CSV format all have the same number of fields no matter if the trailing fields have values in them or not.

Summary

We would like to emphasize that with Google Cloud Platform technologies such as Google BigQuery, Google Compute Engine, and Google Cloud Storage, harnessing value out of Big Data is in everyone's reach both economically as well as from a skill portability perspective. Through the technology partner network Google has created around its Enterprise Cloud offerings, every IT organization will be able to get started with Big Data in the Google Cloud Platform without having to upskill IT resources or even acquire new technologies in most cases.

Special Note: *We did an extensive analysis using the Tableau DeskTop Version, Tableau connecting to BigQuery. Real time to perform queries was a great experience, and query results were produced in a matter of seconds. We were able to accomplish significant analysis on such a large amount of data in a matter of a few hours (traditionally it would have taken us days since data analysis is not the focus of this post, we have not detailed our experience with Tableau).*

##

Reference Links:

Loading Data into BigQuery

Querying Data with BigQuery

Developer Tools: API Overview

Python Application to access Google Cloud Storage from the Command Line

About the Author

Raj Gondhali is a Big Data Principal at Saama Technologies, Inc.
www.saama.com

Contributor

Dena Lawless is the Director of Marketing for Saama Technologies
www.saama.com/data-post/